AARHUS
UNIVERSITY
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Bachelor Project
System Requirement & Testing Specification

by

Daniel Christopher Biørrith, 201909298

Lukas Koch Vindbjerg, 201906015

Bachelor's Thesis
in
Computer Engineering

Supervisor: Kim Bjerge

Aarhus University
Department of Electrical and Computer Engineering

June 15, 2022

Contents

Chapter 1

Introduction

1.1 Purpose

This document provides a description of the functional and non-functional requirements of the Digital Foraminifera Isolation (DFI) system, which in turn provides a description of what the system should to do and how it will perform should be presented. Additionally, based on said requirements, a test specification will be presented.

1.2 Glossaries

Acronym and technical terms	Description
DFI	Digital Foraminifera Identifier, the formal name of the project, which it will be referred to as throughout the report.
CNC system	Computer numerical control system.
Pylon 6	A software provided by Basler. It comes both with a program, Pylon Viewer, which makes it possible to get livefeed from the microscope, as well as an SDK for developing code to control the camera.
SDK	Software development kit
Foraminifera	A singlecelled organism often found in sediment samples.
Sediment sample	Sediment samples are samples consisting of foraminifera and other granular material.
Tray	The observation platform used to place sediment samples on during scanning.
G-code	A highly popular CNC programming language, used to communicate with the controller.
Grbl	The software located on the controller, which interprets the G-code sent.
VS2013	Visual Studio 2013 is an IDE provided by Microsoft, which is used to develop the acquisition software.
DIP	Digital Image Processing
CV	Computer Vision
GUI	Graphical user interface.
ML	Machine Learning
USB	Universal serial bus
OpenCV	An open source CV library for Python.
Scikit-image	A collection of algorithms for image processing, developed for Python.
Tensorflow	An open source CV library in Python, used for creating ConvNets.
JupyterLab	An IDE primarily used to execute .ipynb files.

Table 1.1: List of abbreviations and descriptions thereof

Chapter 2

System introduction and description

2.1 System description

This section will go into a more detailed description of the system, its component and which components are designed to handle which actions. The setup of the system can be seen in figure 2.1. The idea is to use a digital microscope to capture images of the sediment sample. The sample should be spread out such that none of the granular material obscure one another. Hence, we use a black tray - shown on figure 2.1 - as a consistent platform for the samples, to improve consistency. The

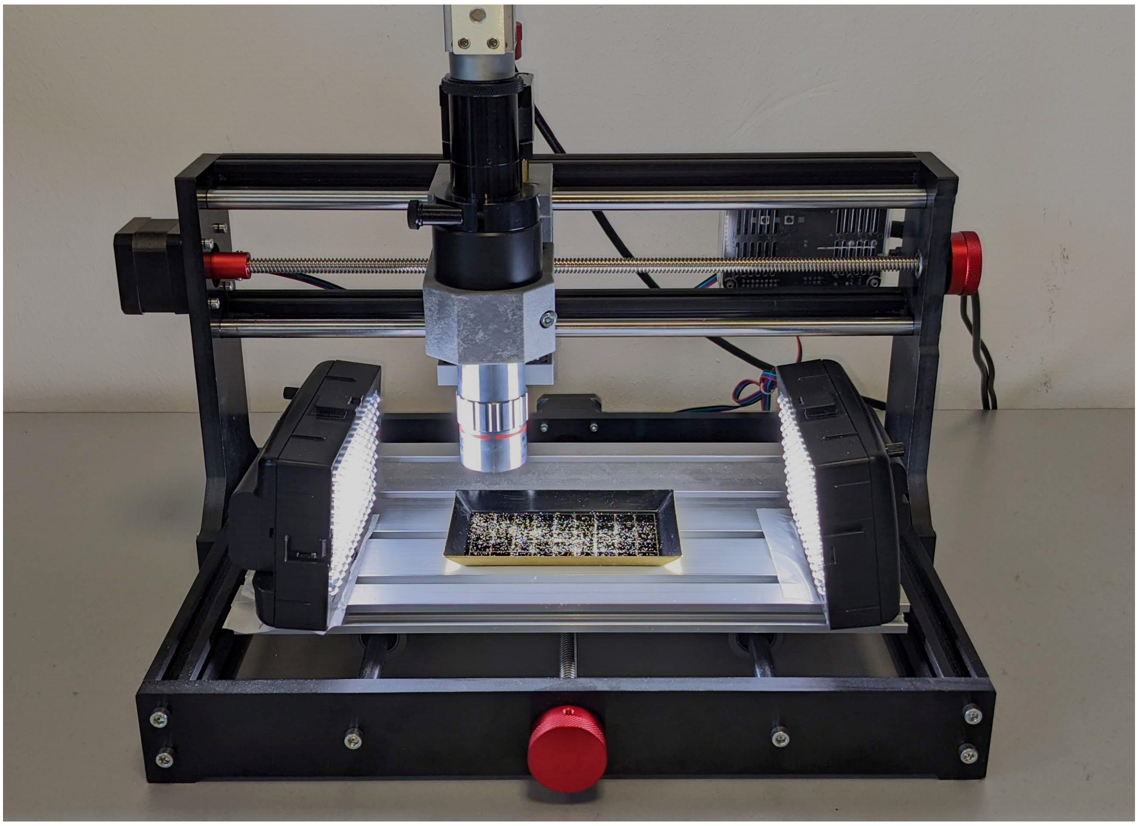

Figure 2.1: Setup of DFI system

microscope only captures a very small area of the tray and can not capture more than a few dozen grains in one image. Consequently, we have to move it across the tray to capture the entire sample. To do this, the microscope has been rigged to CNC-router. This allows us to control the movement of the microscope to traverse any specified area, with periodic stops to allow the microscope to capture clean pictures. The CNC-router is controlled by a control board attached to the setup. Both the microscope and the control board of the CNC-router can be controlled from the users pc. This allows the user to change settings and procedures depending on their intended use. Finally, a pair of LED lights have been mounted to the system to provide the sensor of the microscope with more light, to optimize the process and maximize the information preserved from the real world sample. With the setup initialised, the acquisition process can then be started and the raw images will be stored on the user's PC With all the images acquired, the next process of the system is to isolate individual objects. This will be done using theory and methods from the field of digital image processing. The selected pictures will first be stitched together into one bigger image and then the granular material will be segmented into images which only contain a single object. These will again be stored on the user's pc.
Finally the system will classify these segmented objects using a machine learning model. It will classify an object as either foraminifera or sand. These sets of both foraminifera and sand will likewise be stored on the user's pc in separate directories.

2.2 Assumptions and Dependencies

For our system to function properly we have to make some assumptions that must be true. Hence, we have to depend on some external features. These are mainly the specifications used for the system and some features - both on a hardware and a software level - where the functionalities have been developed by external actors. Therefore, we simply have to rely on these aspects of our system to be working correctly.

D01. The first article that we are dependant on is the physical CNC-machine[1]. We will assume that with any code and implementation we will get a consistent result. This means that we expect that the movement of the motor will be reliable enough for us to successfully stitch together a larger canvas without any missed sports or distortion.

D02. In order to move the CNC-machine correctly we therefore also expect the control board[2] to interpret our software correctly. The way we interact with the CNC-router is through the use of Grbl 1.1f which is an embedded, high performance g-code-parser and CNC milling controller[3][4].

D03. Additionally, the microscope setup[5] is a crucial part of the system's functionality. Here we expect consistency in the images. Any noise or blurriness must stay low enough such that the images contains enough reliable data to be processed in the later steps of the system.

D04. In order to control the microscope setup we take advantage of the Basel Pylon 6 software[6]. This includes the program developed by Basel themselves alongside some provided template code and tools[7][8] which are designed to build applications in C++.

D05. The software we use for the DM system is developed to work on the Windows operating system. Therefore we can not ensure correct functionalities on any other OS and expect that the user will interact with the system with Windows.

D06. In order to compile and execute our software implementation it is a requirement that we use Visual Studio 2013[8] as there are some libraries and extensions that requires us to use this version of the Visual Studio IDE.

D07. The system should be used in stable and well lit environment. The image quality is highly sensitive to movement so if the picture taking process is being performed on an unstable surface this might affect the final image quality which we cannot guarantee will give the same accuracy in the results.

Chapter 3

Requirements

By applying the requirement engineering process[9, p. 101] we have derived the functional- and non-functional requirements of the DFI system. The functional requirements are presented by applying the use-case method[9, p. 125], where each use-case is defined using (i) the natural language specification[9, p. 121] and (ii) the structured specification[9, p. 122]. Furthermore, the non-functional requirements are presented using only the natural language specification.

3.1 Functional Requirements

The following section will present the functional requirements of the system. This will be presented both by some statements regarding how the system should work, as well as with use cases as previously described. Some functionalities that the system must have are described below:

FR01: The motors controls the movement of the microscope.

FR02: The motors must move the microscope systematically over the samples, with slight image overlap, such that it doesn't miss any potential Foraminifera.

FR03: The camera must stand still for a fixed amount of time, depending on the exposure time, providing the microscope enough time to grab an image that is in focus.

FR04: The system must be deployable on a PC on which Windows 11 is installed.

FR05: The images should be stored on the host PC, so that the user may access and review them manually.

Other functional requirements will now be described with use cases.

3.1.1 Actors

In order to understand the use-cases, we first have to define the actors using the DFI system, as they are actuating the use-cases. These actors can be split into two groups; the primary- and secondary actors. The primary actor is the human operator of the system, whereas the secondary actors the samples which contains the Foraminifera. As the hardware and software components are part of the system and not external, these will not be included as actors.

Name, type and description of each actor can be found in table 3.1.

Actor name	**Actor type**	**Actor description**
User	Primary	The manual user of the system. The user is the only primary actor, meaning it's the only actor responsible for starting the different parts of the system.
Sample	Secondary	The sample that is to be analyzed, which contains the Foraminifera. This should be distributed onto the tray as evenly as possible. The sample is a secondary actor as it is necessary for the primary actor to perform certain use cases. More on this later.

Table 3.1: Actors of the DFI system

With this setup, the hardware and software that are part of the system, rather than actors. This also goes hand-in-hand with the dependencies we previously stated, as we now expect these to work, unlike an actor who have to follow the steps.

3.1.2 Use case diagram

With the actors defined, a use-case diagram of the system was developed. The diagram is depicted in figure 3.1.

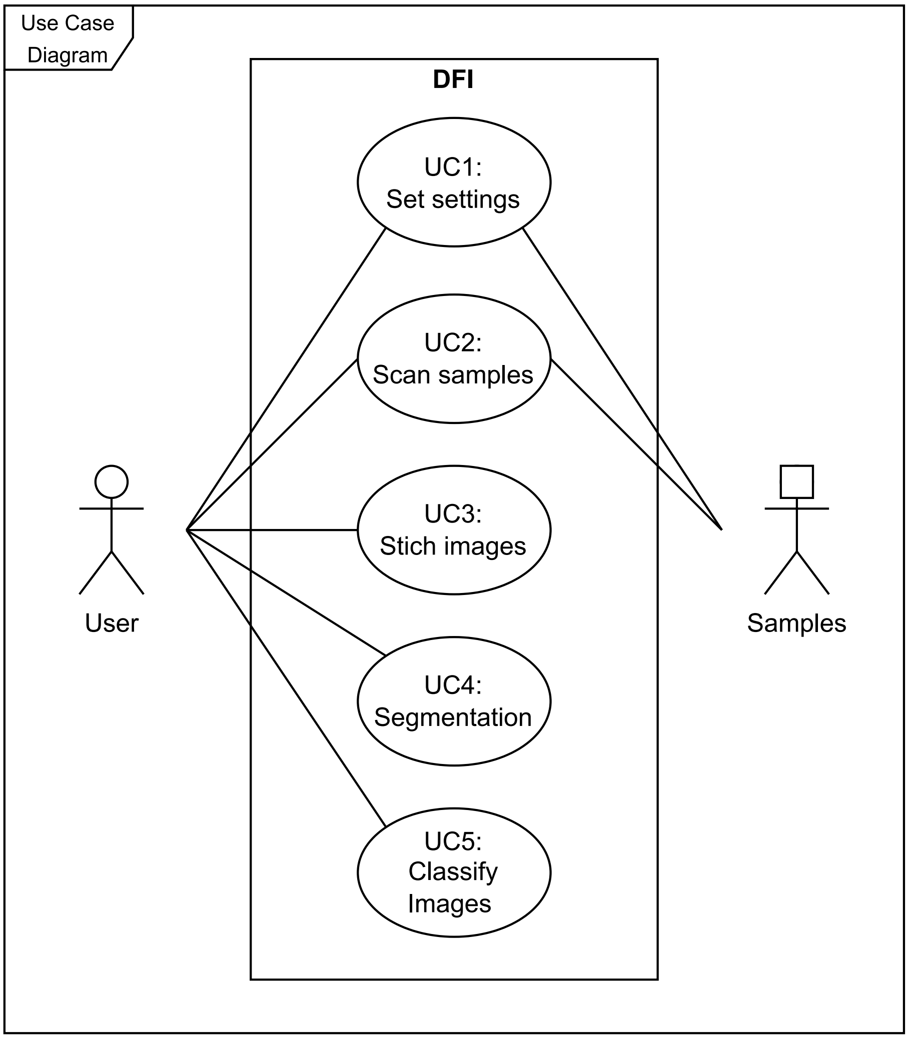

Figure 3.1: UML Use Case Diagram

3.1.3 Use case overview

Here follows an overview of each use case found in the use case diagram, including a brief description of each use case. This can be found in table 3.2.

ID	Name	Description
UC1	Set settings	The actor sets variables, such as exposure time and depth, to their wanted value for optimizing the picture. When the preferred values are chosen, the actor can either save the values or cancel, which would prompt to go back to the previously set values, or default in case none are set prior.
UC2	Scan samples	Following UC1, when the actor considers the system and camera are set up correctly for the desired task, the actor starts the scanning process. This prompts the system to systematically move the camera and capture images of all the samples. These images are then stored in a local directory with an appropriate naming scheme.

UC3	Stitch images	Following UC2, the images needs to be stitched. If the images have been capture, the actor may start the image stitching program with images stored in a local directory on the host machine. After starting the stitching program, it will start stitching all the individual pictures together into one large canvas. When finished, this canvas will likewise be stored on the local machine.
UC4	Segmentation	The actor provides the isolation program with an image, in which the actor wishes for the present object to be detected and isolated. Here, the object will be detected using digital image processing methods. These detected objects will then be isolated into individual picture. Finally, the program will process the images of isolated objects in order to get a consistent format.
UC5	Classify images	The actor provides the identifier program with a directory in which picture(s) are located. The image(s) must be of an isolated object, possibly following UC4. When the actor starts the program, a trained algorithm determines whether provided object is a Foraminifera or not.

Table 3.2: Use-cases natural language specification

3.1.4 Fully dressed use case descriptions

Here, a fully dressed description of each use case follows. The descriptions provide a detailed walk through of each use case, including actors, pre -and post conditions and exceptions that may occur.

3.1.4.1 UC1: Set settings

<table>
<tr><td>Name</td><td>Set settings</td></tr>
<tr><td>Use case ID</td><td>UC1</td></tr>
<tr><td>Actor(s)</td><td>User
Sample</td></tr>
<tr><td>Precondition</td><td>The sample is distributed across the board
Pylon 6 or newer is installed on the operating system.</td></tr>
<tr><td>Main scenario</td><td>1. User starts up the 'Settings' program.
2. A live feed from the camera becomes present for the user to view, along with the parameters that can be changed. This includes exposure time and sensitivity.
Extension 1.1, 1.2.
3. User changes settings to their preferred value.
Extension 1.2, 1.3.
4. User saves the settings, by which the variables will be saved as the parameters the user provided.</td></tr>
<tr><td>Extension</td><td>Extension 1.1:
1. The user quits the program.
2. The variables will keep their prior value.
3. Use case is terminated.
Extension 1.2:
1. The camera is not connected via a viable USB port.
2. The program displays a warning to the user, recommending to restart the program and check the camera connection.
Extension 1.3:
1. The set values exceeds the allowed limit.
2. A warning displays that the value exceed limits. The value of variables will not be changed.</td></tr>
<tr><td>Post condition</td><td>The variables are set to the user's wanted values.</td></tr>
</table>

Table 3.3: UC1 fully dressed description

3.1.4.2 UC2: Scan samples

Name	Scan samples
Use case ID	UC2
Actor(s)	User Sample
Precondition	The sediment sample is spread out on the tray. The tray and camera is manually positioned in the top left of the desired scanning area.
Main scenario	1. The user lines up the camera to the corner of the tray. 2. The user starts the program. **Extension 2.1, 2.2**. 3. The user starts the scanning, which in turn starts the picture grabbing process. 4. The program starts moving the camera and grabs pictures across the tray. 5. The program finishes, by which the pictures are stored in a local directory on the host machine.
Extension	**Extension 2.1:** 1. The camera is not connected via a viable USB port. 2. The program displays a warning to the user, recommending to restart the program and check the camera connection. 3. Use case is terminated. **Extension 2.2:** 1. The controller is not connected via a viable USB port. 2. The program displays a warning to the user, recommending to restart the program and check the controller connection. 3. Use case is terminated.
Post condition	All the images will be stored in a local directory located on the host machine.

Table 3.4: UC2 fully dressed description

3.1.4.3 UC3: Stitch images

Name	Stitch images
Use case ID	UC3
Actor(s)	User
Precondition	UC2 is done.
Main scenario	1. User starts the stitching program. 2. User provides the program with the directory in which the pictures from UC2 are located. 3. The user chooses to starts the process, by which the program starts stitching the pictures together. **Extension 3.1.** 4. The program finished, by which the final picture is created.
Extension	**Extension 3.1:** 1. The pictures cannot be stitched together. 2. No pictures are stitched together. 3. Use case is terminated.
Post condition	All the provided pictures from UC2 are stitched together in a single image that is saved.

Table 3.5: UC3 fully dressed description

3.1.4.4 UC4: Segmentation

Name	Segmentation
Use case ID	UC4
Actor(s)	User
Precondition	UC3 is done.
Main scenario	1. The user starts up the segmentation process. 2. The user provides an image to the program from which they wish to isolate the objects on. 3. The user chooses to start, by which the program starts isolating the objects. **Extension 4.1** 4. The program finishes, by which the images of the objects images are stored in a local directory on the host machine.
Extension	**Extension 4.1:** 1. No objects can be found. 2. A local directory will be created on the local host machine, but no pictures are stored inside of it. 3. Use case is terminated.
Post condition	The objects are isolated and stored in separate image. They're all stored in a local directory on the host machine.

Table 3.6: UC4 fully dressed description

3.1.4.5 UC5: Classify images

Name	Classify images
Use case ID	UC5
Actor(s)	User
Precondition	
Main scenario	1. The user starts the identification program. 2. The user provides the objects that were isolated from UC4, or alternative data. 3. The user starts up the identification process where the program will begin to classify each image by applying our machine learning model. 4. The program will label each object and put it into its respective directory, depending on the classification.
Extension	
Post condition	The images will be stored accordingly to their classification in directories with an appropriate naming scheme of their type.

Table 3.7: UC5 fully dressed description

3.2 Nonfunctional Requirements

Besides the functional requirements, some non-functional requirements were also derived. These help put a constrains on services offered by the system. They are as follows:

NF01 The scanning of the samples must be done within a time-frame of maximum 1 hours 30 minutes.

NF02 The stitching of images must be done within a time-frame of 2 hours.

NF03 The segmentation of an image must be done within a time-frame of 2 hours.

NF04 The classification of the sample images must be done within a time-frame of maximum 2 hours.

NF05 The tray used for the samples must be 9,5 x 5,5 cm in size.

NF06 The classification must have an accuracy of over 85%.

NF07 The communication between camera and the host computer is through a serial connection.

NF08 The communication between controller and the host computer is through a serial connection.

Chapter 4

Testing introduction

This chapter will introduce the terms the testing DFI will build upon. This will help ensure that the system meets the requirements. Thus, testing is build upon the requirements set earlier in this document.

4.1 Acceptance of Test Cases

In order for a test case to be considered accepted, the result of the test case must match the expected result. Each test case will in the end be marked with on of the following expressions:

- **Passed**
  A ✓ will represent passed. It means the test case is accepted.
- **Passed w/ remarks**
  A (✓) will represent passed with remarks. It means the test case is partially accepted but with certain remarks. These remarks will be explained in detail in the 'Comments' column.
- **Failed**
  A ÷ will represent failed. It means the test case is not accepted.

4.2 Preparation prior to testing

Prior to the test cases, the follow statements must be true for them to be viable:

- The physical set-up must be as described in the chapter 2.
- All physical components must be turned on.
- The sediment samples must be distributed manually across the tray.
- All programs must be installed on the PC device and be able to start.

With these done, one is ready to test the system.

Chapter 5

Testing of functional requirements

The following section will specify testing the functional requirements specified Requirement Specification document. The test cases are both of the functionalities and the use-cases. Additionally, test specification of the extensions for each use case will be made.

5.1 Functionalities

In table 5.1, a description of what is to be tested of the is presented.

Functionality requirements			
No.	**Expected result**	**Conclusion**	**Comments and remarks**
1	The motors controls the movement of the microscope.		
2	The motors must move the microscope systematically over the samples, with slight image overlap, such that it doesn't miss any potential Foraminifera.		
3	The camera must stand still for a fixed amount of time, depending on the exposure time, providing the microscope enough time to grab an image that is in focus.		
4	The system must be deployable on a PC on which Windows 11 is installed.		
5	The images should be stored on the host PC, so that the user may access and review them manually.		

Table 5.1: Testing specification of functionalities defined in the requirement document.

5.2 UC1 testing specification

Use case 1: Set settings				
No.	**Action**	**Expected result**	**Result**	**Comment**
1	User starts up the 'Settings' program.	The setting program starts and a live feed from the camera becomes present, along with a UI guiding the user.		
2	The user sets a setting with an input	The setting will be set to the provided input		

Table 5.2: Testing specification of use case 1

5.2.1 UC1 extension(s)

Use case 1 extension 1.1				
No.	**Action**	**Expected result**	**Conclusion**	**Comment**
1	The user quits the program.	The program terminates with variables keeping their prior values.		

Table 5.3: Testing specification of use case 1 extension 1.1

Use case 1 extension 1.2				
No.	**Action**	**Expected result**	**Conclusion**	**Comment**
1	The camera is not connected via a viable USB port.	The program displays a warning to the user, recommending to restart the program and check the camera connection.		

Table 5.4: Testing specification of use case 1 extension 1.2

Use case 1 extension 1.3				
No.	**Action**	**Expected result**	**Conclusion**	**Comment**
1	The set values exceeds the allowed limit	A warning displays that the value exceed limits. The value of variables will not be changed.		

Table 5.5: Testing specification of use case 1 extension 1.3

5.3 UC2 testing specification

Use case 2: Scan samples				
No.	**Action**	**Expected result**	**Conclusion**	**Comment**
1	The user starts the program.	The program will start, providing the user the option to start scanning.		
3	The system performs the grabbing of images.	Images of the tray are acquired in a systematic manner and stored in a local directory.		

Table 5.6: Testing specification of use case 2

5.3.1 UC2 extension(s)

Use case 2 extension 2.1				
No.	**Action**	**Expected result**	**Conclusion**	**Comment**
1	The camera is not connected via a viable USB port.	The program displays a warning to the user, recommending to restart the program and check the camera connection.		

Table 5.7: Testing specification of use case 2 extension 2.1

Use case 2 extension 2.2				
No.	**Action**	**Expected result**	**Conclusion**	**Comment**
1	The controller is not connected via a viable USB port.	The program displays a warning to the user, recommending to restart the program and check the controller connection.		

Table 5.8: Testing specification of use case 2 extension 2.2

5.4 UC3 testing specification

Use case 3: Stitch images				
No.	**Action**	**Expected result**	**Conclusion**	**Comment**
1	User starts up the Stitching program.	The program will start, asking the user to provide a directory.		
2	User provides a directory and starts the stitching process.	The images in the provided directory are stitched together to one large image.		

Table 5.9: Testing specification of use case 3

5.4.1 UC3 extension(s)

Use case 3 extension 3.1				
No.	**Action**	**Expected result**	**Conclusion**	**Comment**
1	The images cannot be stitched together	A warning message appears, telling the user the images could not be stitched together.		

Table 5.10: Testing specification of use case 3 extension 3.1

5.5 UC4 testing specification

Use case 4: Segmentation				
No.	**Action**	**Expected result**	**Conclusion**	**Comment**
1	User starts up the Segmentation program.	The program will start, asking the user to provide a directory.		
2	User provides a directory and starts the segmentation process.	A directory will be created, in which all segmented objects are stored in their own image.		

Table 5.11: Testing specification of use case 4

5.5.1 UC4 extension(s)

Use case 4 extension 4.1				
No.	**Action**	**Expected result**	**Conclusion**	**Comment**
1	No objects are found on the image.	A warning message appears, telling the user no objects could be found.		

Table 5.12: Testing specification of use case 4 extension 4.1

5.6 UC5 testing specification

Use case 5: Classification				
No.	**Action**	**Expected result**	**Conclusion**	**Comment**
1	User starts up the Classification program.	The program will start, asking the user to provide a directory.		
2	User provides a directory and starts the classification process.	A directory, which contains sub-directories for the groups of classes, will be created. Objects will be stored in a respective folder, depending on what they're classified as.		

Table 5.13: Testing specification of use case 5

5.7 Result

The result of each case will be summarized in a table 5.14.

Use case testing summary		
UC No.	**Results**	**Comments**
UC1		
UC2		
UC3		
UC4		
UC5		

Table 5.14: Summary of use case testing.

Chapter 6

Testing of non-functional requirements

The following section provides an overview of how the non-functional requirements will be tested.

The tests of non-functional requirements are specified in table 6.1. Not all non-functional requirements are included. The ones not included are: NF05, NF07 and NF08, as are part of the design instead of testing.

Use case testing summary				
ID	**Requirement**	**Testing**	**Result**	**Comments**
NF01	Acquiring images of the whole tray must finish within a time-frame of 1 hour 30 minutes	The acquiring program will be timed.		
NF02	The stitching of images must be done within a time-frame of 2 hours	The stitching program will be timed.		
NF03	The segmentation of an image must be done within a time-frame of 2 hours	The segmentation program will be timed.		
NF04	The classification of images must be done within a time-frame of 2 hours	The classification program will be timed.		
NF06	The classification must have an accuracy of over 85%.	The model will be tested in a test-set of data.		

Table 6.1: Summary of use case testing.

Bibliography

[1] Genmitsu Industries LLC. "Original genmitsu 3018-pro." (2022), [Online]. Available: https://genmitsu.com/ (visited on 03/29/2022).

[2] Desktop CNC Supply. "Woodpecker cnc control board v3.4." (2022), [Online]. Available: https://www.3018nc.com/product/3018-pro-cnc-router-control-board/ (visited on 03/29/2022).

[3] Grbl CNC controller. "Grbl." (2021), [Online]. Available: https://github.com/grbl/grbl (visited on 04/04/2022).

[4] reprap. "G-code." (2022), [Online]. Available: https://reprap.org/wiki/G-code (visited on 04/04/2022).

[5] A. Kathigamanathan, R. K. H. Thomsen, and J. E. E. Sømod, "Udviklingen af et fuldautomatisk system til artsbestemmelse af foraminiferer ved brug af neurale netværk," Dec. 2020.

[6] Basler AG. "Pylon 6.0.1 camera software suite windows." (2022), [Online]. Available: https://www.baslerweb.com/en/sales-support/downloads/software-downloads/pylon-6-0-1-windows/ (visited on 03/29/2022).

[7] ——, "Pylon sdk samples manual." (2021), [Online]. Available: https://www.baslerweb.com/fp-1615186793/media/downloads/documents/users_manuals/AW00148804000_pylon_SDK_Samples_Manual.pdf (visited on 03/29/2022).

[8] ——, "How to build basler pylon c++ applications with free microsoft visual studio." (2017), [Online]. Available: https://www.baslerweb.com/fp-1508418766/media/downloads/documents/application_notes/AW00064406000_Application_Note_Build_pylon_C_Apps_with_Free_Visual_Studio_IDE.pdf (visited on 03/29/2022).

[9] I. Sommerville, *Software Engineering, Global Edition*, 10. edition. Pearson Education Limited, 2016, ISBN: 9781292096148. [Online]. Available: https://books.google.dk/books?id=W_LjCwAAQBAJ (visited on 03/29/2022).

[10] D. C. B. Lukas Koch Vindbjerg. "Appendix." (2022), [Online]. Available: https://github.com/Biorrith/Bachelor-project/tree/main/References (visited on 04/11/2022).